(12) UK Patent Application (19) GB (11) 2 149 944 A

(43) Application published 19 Jun 1985

(21) Application No 8427016

(22) Date of filing 25 Oct 1984

(30) Priority data
(31) 551125 (32) 14 Nov 1983 (33) US

(71) Applicant
Softnet Incorporated (USA–Massachusetts),
53 Dean Road, Weston, Massachusetts 02193, United States of America

(72) Inventors
Lance E Hansche
Neil J Colvin

(74) Agent and/or Address for Service
Withers & Rogers,
4 Dyer's Buildings, Holborn, London EC1N 2JT

(51) INT CL^4
G06F 13/38 H04L 9/02

(52) Domestic classification
G4A AP
H4P DCSX

(56) Documents cited
GB A 2124856 EP A1 0090771 EP A1 0008033
GB A 2122777 EP A1 0089876
GB A 2020513 EP A1 0089087

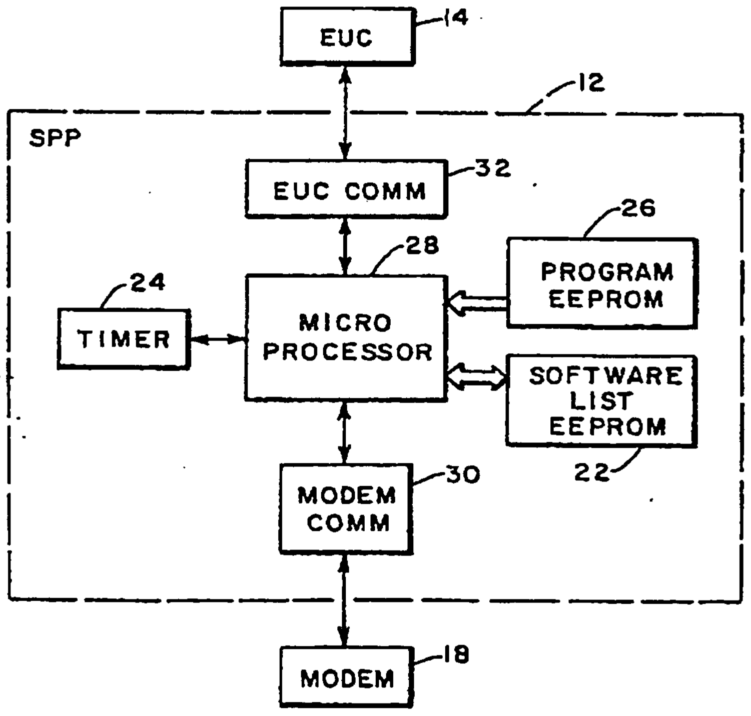
(58) Field of search
G4A
H4P

(54) Software distribution

(57) Software is protected against unauthorized copying. The software is encrypted in a host computer and then transferred to the end-user computer EUC after it is registered in the software protection computer SPP. Portions of the transferred software are encrypted using a unique encryption key. Each copy of a software package generated by the host computer is a unique encrypted version of that software package, which when it is run on the end user's computer and encounters an encrypted portion of itself, suspends normal execution and transfers the encrypted portion to the software protection computer. This computer then decrypts the encrypted portions of the code and returns the decrypted portions to the end-user computer where that code is itself executed or allows execution of the program of which it is a part to continue. The software package is received along with a particular user's decryption key, stored in the software protection computer.

SPP BLOCK DIAGRAM

FIG. 2



GB 2 149 944 A

The specification as filed includes a computer program which is not here reproduced; it may be inspected in accordance with Section 118 of the Patents Act 1977.

SYSTEM BLOCK DIAGRAM

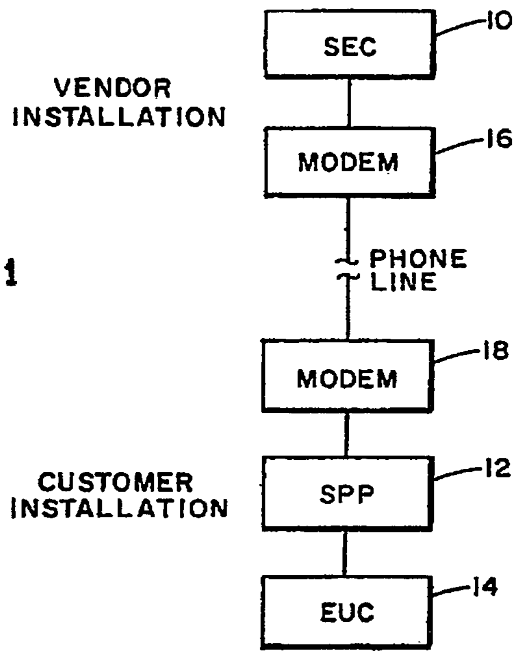


FIG. 1

SPP BLOCK DIAGRAM

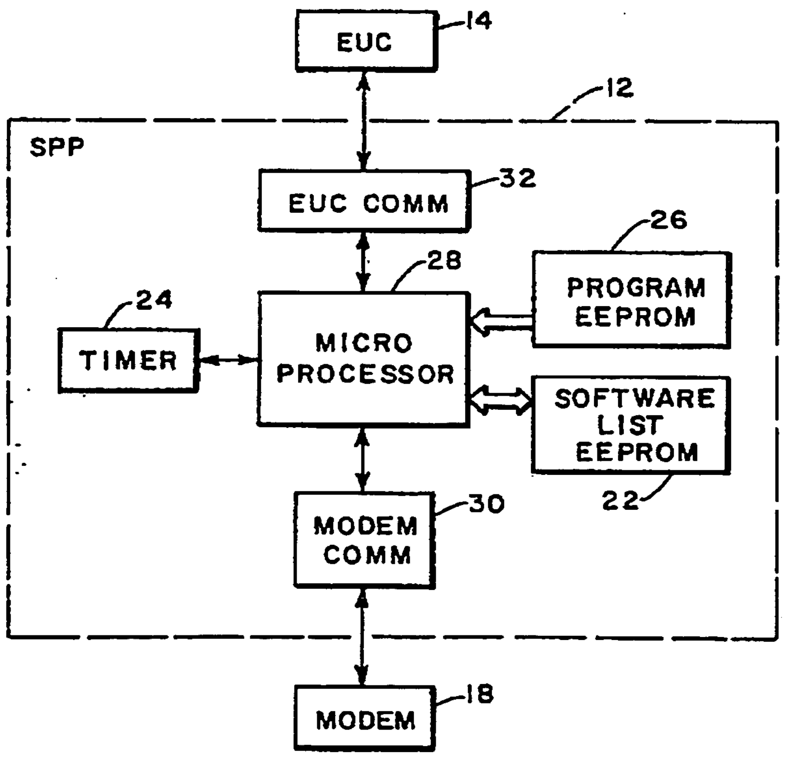


FIG. 2

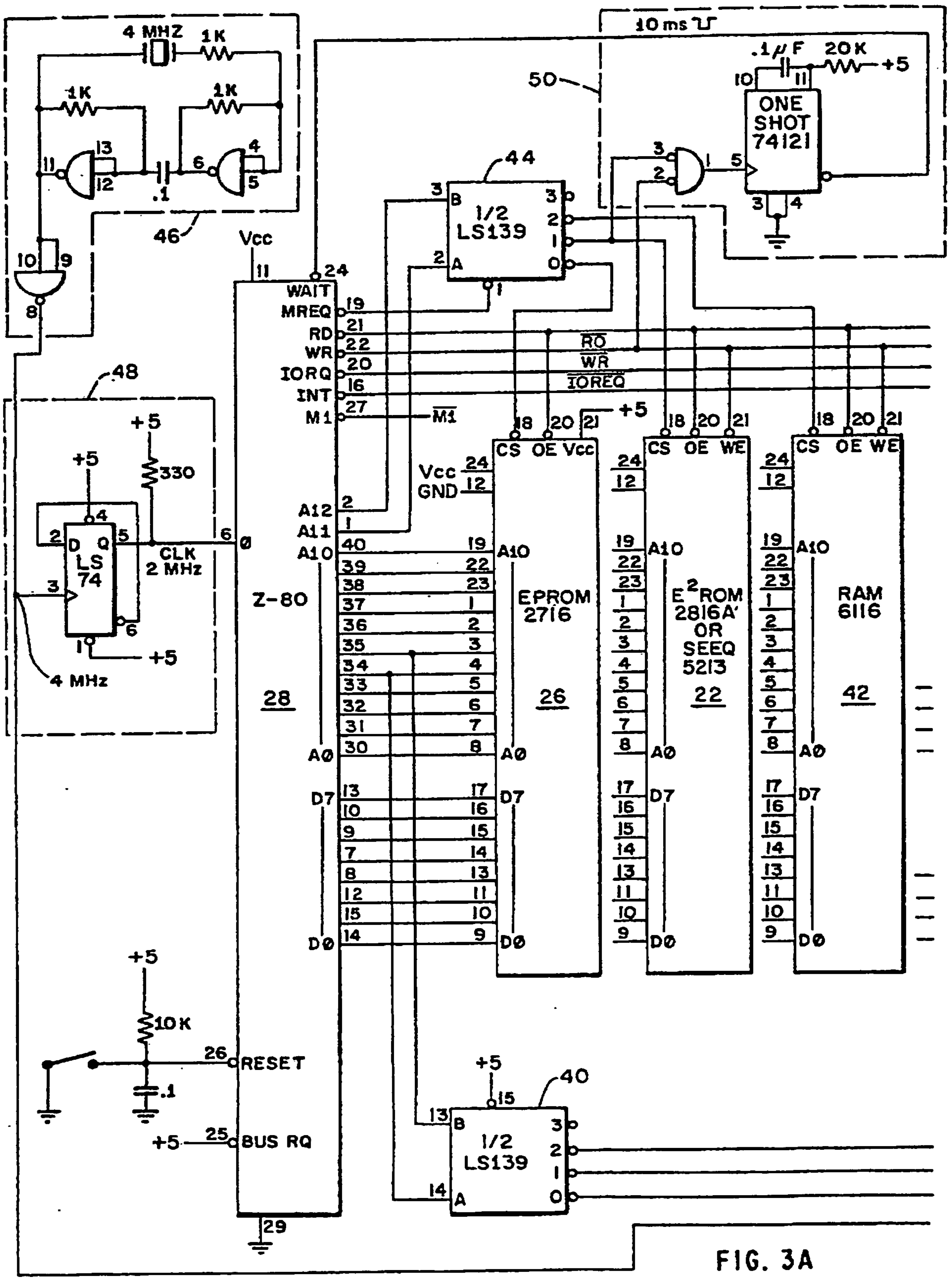

4 MHZ
1K
1K
1K
.1
46
48
+5
330
LS 74
CLK 2 MHz
4 MHz
Vcc
Z-80
28
WAIT
MREQ
RD
WR
IORQ
INT
M1
A12
A11
A10
A0
D7
D0
RESET
BUS RQ
10K
44
1/2 LS139
50
10 ms
.1μF
20K
ONE SHOT 74121
RD
WR
IOREQ
M1
EPROM 2716
26
E2ROM 2816A' OR SEEQ 5213
22
RAM 6116
42
CS
OE
WE
Vcc
GND
40
1/2 LS139

FIG. 3A

MEMORY		I/O	
EPROM	0–7FF	CTC	10–1F
EEROM	800–FFF	CLOCK	20–2F
RAM	1000–1FF	SIØ	00–DF

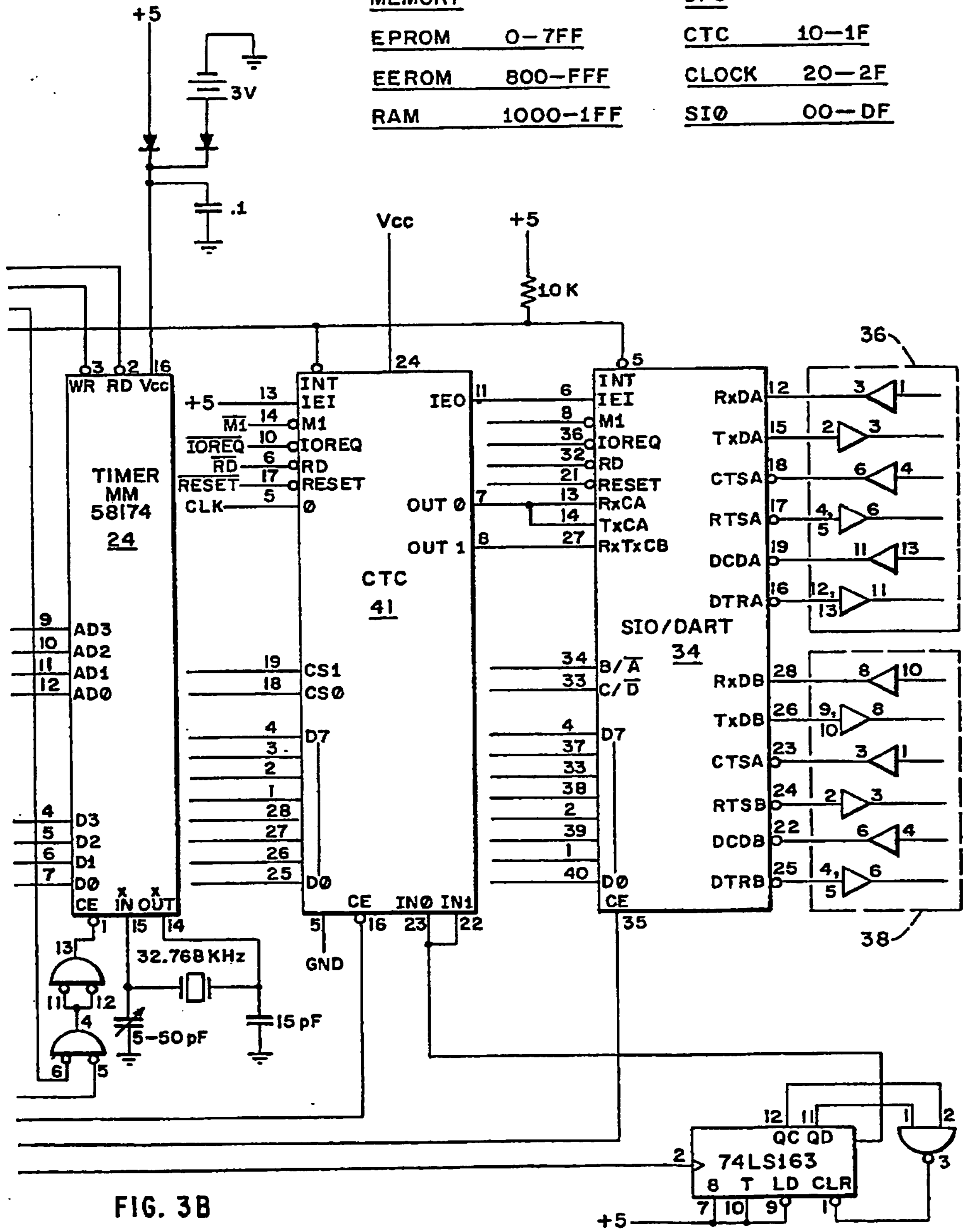


FIG. 3B

SPECIFICATION

Software distribution system

This invention relates to electronic software distribution and more particularly to a software distribution system in which the distributed software is protected against copying.

Over the past few years, the growth of the software industry has been enormous, and as more and more people purchase personal computers, the industry is expected to continue to grow rapidly. For the most part, purchased software changes hands from a mail order or retail vendor to a customer in some physical form such as a tape, disk or even a printed listing of code. Such physical distribution has resulted in a number of problems with respect to both the mode of distribution and customer servicing as well as with the rights of the creators and publishers of the software which is sold. Principal among the problems is that a large percentage of the software which is sold ends up being illegally copied. Frequently, a purchaser of software will "lend" his copy of the software to a friend who makes a copy for himself.

The most obvious result of this unauthorized copying is that the profits of the creator and publisher of the software (who probably have a copyright in the software) are greatly reduced. To make up for these lost profits, the price of the software is maintained at a high level. This sustained high price unfortunately produces an even greater incentive to illegally copy.

Copyright protection, which does provide the creator and publisher of software with legal recourse against the person making the unauthorized copies has, in fact, afforded little or no relief from the problem of copied software. As the copies are often made by individuals for their own use, large-scale policing of such copying is virtually impossible. On rare occasions, a copier having a large copy resale operation can be caught, but by the time he is caught, many unprotected copies usually already have been distributed. Furthermore, the advent of software rental shops has further limited the copyright owner's ability to protect his rights in the software he owns.

Another problem frequently encountered with software sold over the counter is the need to later distribute revised copies to add new features or to fix errors or "bugs" present in the software. These bugs appear despite rather substantial testing that is performed before a software package is put on the market. These bugs are particularly prevalent in software which has recently entered the market. In order to correct any errors which do appear in the software, a software publisher must recall the disk or tape which contains the faulty software. The problem with correcting errors in this manner is that the software is out of the hands of the purchaser for a number of days, if not weeks, while the exchange and correction take place. Finally, the cumbersome nature of this system discourages the user's updating of his software which often leaves a bad impression of the software publisher's products in the field.

In order to combat the illegal copying of software, the software industry has taken a number of precautions. The various approaches fall under three categories: media protection against copying, use of read-only media and processor serialization.

Media protection against copying refers to making some unique version of the medium containing the software. One type of media protection involves the use of variable-pattern diskettes. Variable-pattern diskettes, however, do not offer a practical solution to the software copying problem since these diskettes depend on a soft format diskette drive and they are vulnerable to memory copy if the entire program is loaded at once. Furthermore, such variable-pattern diskettes can only be used in a small percentage of the drives currently on the market. Therefore, the software distributed on such diskettes can only be offered to a rather small percentage of the market. Finally, physical alteration of the media, usually by forcing hard errors on the media checked for by the software itself, has been used. This method fails in that hardware checks in the software can be located and neutralized in the software itself.

Another type of media protection against copying involves the use of an operating system override. Such a protection scheme depends on a rather unique operating system which prevents copying of diskettes. The use of an operating system override, however, has not proven to be the answer to the problem either since the altered operating system must be tailored to the particular controller chip of the computer on which it is operating, and the operating system override cannot support use with standard operating systems currently on the market. In addition, any operating system override is vulnerable to an algorithmic solution or "cracking". One variation on the operating system override scheme has the software employ features of the hardware, circumventing the operating system, to check areas on the storage media which the operating system cannot reach. This method can also be defeated by being neutralized in the software itself.

A third type of media protection against copying involves the use of segmented programs in conjunction with variable-pattern diskettes and/or an operating system override. The use of such segmented programs of necessity requires some type of a segment loader to read in the various segments when required. This results in very slow response from a computer utilizing such segmented programs. Furthermore, any loader routine for reading in segmented programs is vulnerable to algorithmic solution. In addition to the problems stated above, these media protection devices have generally been perceived as being user-unfriendly, and since it is not possible to make a legitimate backup copy, such protection schemes are not in wide use.

Another possible solution to the problem of software copying involves the use of read-only media to store the software. Among the read-only type media which may be used are ROMs and laser cards. The problem with the use of such read-only media is that any software update can only be done by replacing the media itself, and therefore any software update becomes rather expensive. Moreover, there is no legitimate backup for any media failure since a backup copy cannot be created. Finally, with the use of read-only media, added expenses are incurred by the user, since a particular type of reader for that media must be purchased at great expense to the user (with the exception of ROMS) with that user gaining no significant additional value.

The third type of protection, processor serialization, has also not proven to be a very effective means of protecting software. The reason for the ineffectiveness of this mode of protection is that processor serialization requires either the compliance of all computer manufacturers or publisher-supplied hardware which comes with the software package to provide the serialization. In addition, this protection technique adds no value to the computer to compensate for the cost, and there is no benefit to the manufacturer for complying with a processor serialization scheme. Finally, since serialization involves a passive device, it is easy to defeat the serial number check in the software itself.

In light of the problems encountered with the above-described currently existing protection schemes, it appears that illegal sales or copying cannot be stopped altogether; it can only be made more difficult. The ultimate goal of any protection scheme therefore is to make the cost of cracking the protection scheme comparable to or preferably greater than the cost of purchasing the software. In order to make cracking costs greater than the purchase price of the software, the protection scheme must not employ an algorithmic easily solved. In addition, any add-on hardware must be of a low cost nature, and must be compatible with the machines of a majority of the major computer manufacturers.

Therefore, it is a principal object of the present invention to provide a software distribution system which can protect software from being copied.

Another object of the present invention is to provide a software distribution system in which software is encrypted using a virtually indecipherable encryption key.

Still another object of the present invention is to provide a software distribution system in which each copy of the distributed software is protected by a unique encryption key.

Yet another object of the present invention is to provide a software distribution system in which each copy of a program is organized in a unique pattern to frustate comparison.

A further object of the present invention is to create a software distribution system in which revisions in software can be easily distributed.

These and other objects of the invention are achieved by an electronic software distribution system in which distributed program copies are uniquely associated with specific hardware to which the end user's computer must be connected. A central computer facility operated, for example, by a software vendor, contains storage capacity for a library of available programs. Auxiliary Software Protection Processors (SPP) are issued to the users. Each SPP is electrically connected to the user's computer and electronically interconnected with the central facility, for example, via a modem-interfaced phone link. Each SPP is equipped with a unique number code referred to as the package encryption key (PEK) which is recorded at the central facility. The PEK can be factory loaded or down-loaded (via the communications link) to the SPP from the central facility. The software distribution system of the present invention embodies two distinct unique operations: (1) software preparation and delivery and (2) software execution in the user's computer.

In the preparation/delivery phase, when a user orders software from the central facility, the facility first looks up the PEK for that user's SPP and selects an available registration index number (RIN) which will be unique for that user's copy of the software package. The central facility then prepares the unique user copy of the ordered program by encrypting passages of the program selected by the central facility in a manner such that a given algorithm operating on a key specified by a combination of the PEK and RIN and an encrypted passsage will yield the original unencrypted version ("plaintext") of such passage. The encrypted version of the ordered program (which is encrypted only in a subset of its parts or modules) is then transmitted to the user along with a control block containing the RIN. The control block is stripped off and the RIN stored in the user's SPP while the transmitted program copy (with its encrypted passages) is stored in the user's computer system on user-selected media.

In the software execution phase of operation when the user desires to run the program, the initial instructions in the program check the specific RIN in the SPP associated with that program copy. If the RIN is okay, normal execution proceeds until an encrypted passage is encountered. The user's computer then executes a call to the SPP in which the encrypted passage is decrypted algorithmically in the SPP by use of the key specified by the PEK and RIN. The decrypted passage is returned to the user's computer. If the passage is properly decrypted, normal program execution resumes until another encrypted passage is encountered. In the preferred embodiment, these passages may actually be software instructions as well as data.

Time-limited authorization is implemented by means of a real-time clock or counter embedded in the SPP which, for example, erases or alters the software-specific RIN after a trial period or rental term. Since the unique user copy of the selected software cannot run properly unless an SPP with the correct PEK and RIN is engaged with the user's computer system, the software package would therefore be disabled.

This specification includes an Appendix consisting of two parts containing 51 pages of annotated program listings.

The invention will now be described by way of example with reference to the accompanying drawings, in which:

Fig. 1 is a system block diagram showing the various components involved in the transmission of information in the system of the present invention;

Fig. 2 is a block diagram showing the communication interaction of the various components of the system at the user's location; and

Fig. 3 is a circuit diagram of the software protection processor of Fig. 2.

The software distribution system of the present invention provides a means for a vendor to sell software to a vendee while providing protection against copying that software. As shown in Fig. 1, the software distribution system of the present invention includes three computers—a host computer called the Software Encryption Computer (SEC) 10, a software protection computer designated the Software Protection Processor (SPP) 12 and the End-User Computer (EUC) 14. Of these computers, the SEC 10 is owned and operated by the vendor while the SPP 12 and the EUC 14 are owned by the customer and located at a customer installation. The software which is purchased by the customer is transmitted from the SEC through a communication system such as phone lines, a local area network or a cable system. In the preferred embodiment, the software is received by the SPP 12 which transfers the software to the EUC 14 for storage. When the software is transmitted over phone lines, a modem 16 at the vendor installation and a modem 18 at the customer installation are required for sending and receiving the software.

The word "encrypt" is used in this application to indicate a process of taking original code and disguising it so that it is unintelligible. On the other hand, the word "decrypt" is used in this application to describe the reverse process, namely transforming disguised, unintelligible code back to its original form or "plaintext" in the vernacular of cryptography.

The SEC 10 is a central computer facility located at a vendor site or operated under the control of the vendor. The SEC 10 maintains a library of software available for distribution. Each time a software sale is made, the SEC 12 encrypts the copy of the software before transmitting it to the vendee or user. Each copy of software is encrypted in a unique fashion. This is true even if two copies of the same piece of software are transmitted to the same user.

Once the copy of software has been encrypted in preparation for sale, the copy of the software is transmitted by the SEC 10 via the vendor modem 16 to the vendee modem 18 which is connected to the SPP 12. The SPP 12 is a self-contained decryption computer capable of retaining unique control information for each software package purchased by a customer. The SPP 12 has two major functions. The first of these is to confirm the customer's validity and to register control information for any software package sold to that customer. The second is to decrypt any encrypted portions of software received from the EUC 14 which permits that software program to continue operation in the EUC 14. Hence, unless the SPP 12 is engaged, software distributed by the distribution system will not operate in the EUC 14. Although the SPP 12 has been described as communicating with the SEC 10 through a modem 18, the SPP 12 may also contain or interface with communication devices such as a local area network or a cable system. The SPP 12 may also be contained within the user's EUC 14 as well.

The third computer in the preferred embodiment of the present invention, the EUC 14, is a customer owned or operated computer. This computer may be a home computer, personal computer, small business computer or a large main frame computer. All software purchased by a customer is designed for operation on his particular EUC 14.

In operation, before any software may be sold, the customer must purchase a modem/SPP unit and its associated communication software in order to make use of the software distribution system of the present invention. Each SPP 12 has its own unique Package Encryption Key (PEK). The purchased modem/SPP unit is then connected to the customer's EUC 14, and it is simply left in place until the customer wants to purchase software. In the preferred embodiment of the system of the present invention, the customer wishing to purchase software connects his modem/SPP with the system's SEC 10 via telephone. The modem/SPP 12 passes its unique identification code (prefereably in encrypted form) to the SEC 10 to confirm the identification and the legitimate status of the customer. The SEC 10 then generates lists of available software packages along with prices and terms of sale. These prices and terms of sale (usually credit card authorization) must be agreed upon before a transaction actually occurs. Once the customer has met the terms of the sale, the SEC 10 creates a unique copy of the specified software package, and this package, which also contains encrypted security control information, is transmitted through the customer's modem/SPP into his EUC 14. The preparation of the unique copy is accomplished by encrypting selected passages of the software. First, the SEC looks up the unique PEK for the user's SPP. Next, the SEC selects an available Registration Index Number (RIN) specific to the user's software copy. Passages are encrypted in a manner such that they can be decrypted by the SPP using its PEK modified by the package-specific RIN.

When the EUC 14 begins to receive a unique copy of a specific software package, the EUC 14 sends the control information block which arrives first to the SPP 12 for registration. Included in this control information is the encrypted Registration Index Number (RIN) which is decrypted by the SPP 12 and stored in its memory. After the control information has been decrypted by the SPP 12, the remainder of the transmission, the encrypted software package itself, is then passed through the SPP 12 to the customer's EUC 14 for storage on user-selected media. Each time the customer runs software purchased from the SEC 10, his SPP 12 must also be connected and that SPP 12 must be the same SPP 12 which was used when purchasing the software initially. If either of these conditions is not met, then the software will not operate on the EUC 14 because the PEK and the RIN for decrypting any particular software package are only stored in the SPP 12 which was used for purchasing that software.

The two phases of operation are summarized in the following Tables I and II.

TABLE I

Software Preparation and Delivery Phase

1. User with modem/SPP calls SEC.
2. SEC verifies SPP identification number.
3. User selects software from menu.
4. SEC looks up PEK for user's SPP.
5. SEC selects available RIN for user selected software.
6. SEC encrypts selected passages of software in a manner such that they can be decrypted by SPP by algorithmically combining encrypted passage with key generated by modifying PEK with RIN.
7. SEC transmits control block with encrypted version of RIN, followed by software with encrypted passages.
8. EUC passes control block to SPP.
9. SPP decrypts and stores RIN in its memory.
10. EUC stores software with encrypted passages on disk or other media.

TABLE II

Software Execution Phase

1. EUC loads program off disk or other media.
2. Initial module of software tests decryption by sending data to SPP.
3. SPP looks up corresponding RIN and decrypts data with key formed by modifying PEK with that RIN.
4. Software tests returned data and halts execution if data are incorrect.
5. Normal program execution until encrypted passage encountered.
6. At encrypted passage, software jumps to a decryption module which transfers data or instructions to SPP and gets decrypted data or instructions in return.
7. Resume normal execution until next encrypted passage.

The Software Protection Processor (SPP) 12 is the heart of the software distribution system of the present invention since it is the SPP 12 which provides intelligible code to the EUC 14. As shown in Figs. 2 and 3, non-volatile read/write memory 22 is provided in the SPP 12 for storing a valid software list. This non-volatile read/write memory may be implemented in an electrically erasable programmable read only memory (EEPROM) so that the list can be updated with each purchase. The EEPROM 22 will also include a publicly accessible serial number and the PEK. In the preferred embodiment, a clock/timer 24 is also included in the SPP 12 to implement time-limited authorization so that software can be used on a trial or approval basis or rented for a certain predetermined allotted time. The clock/timer 24 is provided with a battery backup. By using such a clock/timer 24 the current time will be updated with every connection to the SEC 10. If there is no battery backup and power to the clock/timer 24 is lost, it is necessary to reconnect to the SEC 10 before any rented software can be run. In addition to the non-volatile read/write memory mentioned above, the SPP 12 will also include a non-volatile read-only memory (ROM) 26 for storing the SPP's operating program. An illustrative operating program in Z-80 assembly language is given in Appendix Part I. If it is desired to provide for later update of the SPP's operating program, however, then an EEPROM can be substituted for the ROM 26 which contains the operating program.

The SPP 12 also includes a Z-80 microprocessor 28 which controls the functioning of the SPP 12. This microprocessor 28 will communicate with both the SEC 10 through modem 16 and with the EUC 14. Appropriate communication interfaces 30, 32 (Fig. 2) are provided between the microprocessor 28 and modem 18 and between the microprocessor 28 and the EUC 14, respectively. These communication interfaces include a dual asynchronous receiver transmitter (DART) 34. The DART 34 communicates with the EUC 14 and the SEC 10 through lines 36 connected between the EUC 14 and the SPP 12 and through lines 38 connected between the SEC 10 and the SPP 12. The DART 34 is linked to the microprocessor 28. Input/output addresses are decoded by circuit 40. A baud rate generator 41 is also included for appropriately matching the modems 16 (Fig. 1) and 18.

The microprocessor 28 preferably includes its own working random access memory (RAM) and it has the ability to execute a program out of either EEPROM. RAM 42 is provided as working storage for microprocessor 28. This RAM 42, as well as EEPROMS 22, 26 are linked to the microprocessor 28. Memory addresses are decoded by circuit 44.

Clock circuts 46, 48 drive microprocessor 28 as well as the baud rate generator 41. A lOmSec delay circuit 50 is also connected to the microprocessor 28 which introduces a delay whenever a write operation is directed to EEPROM 22.

In the preferred embodiment, the modem 18 is included in a "black box" with the SPP. This modem 18 takes data from the microprocessor 28 and transmits it over phone lines, and the modem 18 receives data transmitted over the phone line and passes it on to the microprocessor 28. While all of the above elements of the SPP 12 have been described as individual components, most, if not all, of these functions may be implemented on a single chip or small number of single chip microcomputers.

Another aspect of the present invention which requires special consideration is the Package Encryption Key (PEK) which is created for each customer and his SPP by the SEC 10. This key will be rather large, preferably on the order of 256 bits. Some or all of the bits of the PEK will be used to perform one or more operations on a section of the code having a corresponding number of bits.

For example, if a key of 256 bits is used, the SEC 10 will select portions of the program to be encrypted which also have 256 bits. An operation, such as exclusive ORing (EOR) the two 256 bit codes, is then performed and the resulting 256 bits is inserted into the program at the position from which the selected 256 bits were removed. It is this encrypted version of the software package which is sent to a customer. In order to decrypt this code, the SPP 12 will perform a reverse operation using the 256 bit key and the encrypted 256 bits. In the case where the original operation was an EOR, the reverse operation is also an EOR. The specific key which is assigned to each customer will be stored in the SEC 10 and this key will be used by the SEC 10 when creating each encrypted version of software.

The valid software list which is maintained by the SPP 12 in EEPROM 22 includes an RIN for each entry into the valid software table. This RIN points to a location in the PEK. For example, if a one byte RIN (0-255) in the valid software table for a specific software package is 108, then the program's encryption will be performed using a key which begins at the l08th bit of the customer's PEK. In one embodiment, as each program is sent to the EUC 14, it will be given the next consecutive available RIN for the PEK. In other words, the first program in the valid program table will be given a one byte RIN of 1 into the PEK, the second program sent to the SPP's valid software list will be assigned an RIN of 2 for the PEK and so on. The assigned keys will remain the same size since the ends of the PEK are simply "wrapped around" so that the new end of the PEK is the bit immediately preceding the beginning bit of the PEK.

To summarize, the actual encryption key is a function of the user-specific PEK and the software-specific RIN. The RIN, in this embodiment, simply designates a starting location in the PEK. Other means of combining RIN and PEK to obtain the software-specific encryption key are possible.

Besides encrypting software with a unique PEK/RIN key, the software distribution system of the present invention will provide additional safeguards against copying. For example, since most programs are constructed from small, interrelated modules, portions of each module may be separately encrypted by the SEC 10. These modules may then be linked together by a linkage editor which basically keeps a list of the beginning and end addresses of all modules. When an end of a module is reached, a jump command to the beginning of the next appropriate module may then be put into effect. In this manner, all the modules are tied together. In fact, once such modules are linked in this fashion, the individual modules lose their identity and the program appears to be monolithic. Therefore, to further complicate any attempt to copy software, the software distribution system of the present invention may scramble the order of the modules on a random or other basis. In this way, any person gaining access to two copies of the same encrypted software package sold by the SEC 10 will not be able to locate the sites of encryption by simple comparison.

A Concrete example of program encryption and module randomization is presented in Part II of the appendix. Five sample modules are incorporated in a program called "MAIN1". The program is designed to run on a MSDOS system such as that used on the IBM PC. The unencrypted object code for the program is stated in hexadecimal digits on pages 1-13 of Part II of the appendix. To prepare this software for delivery, a special "security control module" (pages 17-19) is added to handle all calls to the SPP. The security control module acts like a subroutine. Actually, this subroutine engages the "subprogram" in the SPP to decrypt the encrypted passages. To illustrate an encrypted passage, special print data (a part of the software) is presented in connection with modules 1 and 4. As shown on page 16 of Part II, two sets of "external character" data are created namely "*messg1" and "*messg4" in place of the plain text version "This is" module 1 or 4, respectively. (See page 16, Part II.) Before encryption, the print data resides correctly in program memory beginning at hex location 2762 (page 12, Part II). After encryption, the first eight bytes of the print data for modules 1 and 4 is encrypted as shown for module 1 in locations 2762-2769 (page 31, Part II). The encryption was performed by exclusive ORing. The original eight bytes (representing "This is" with the 64 bit (eight bytes) PEK "AAAAAAAAAAAAAAAA". In binary this nonrandom PEK is "1010 ... " Thus the even/oddness of the RIN determines whether the decryption key starts with "0" or "1". The encrypted code on page 31 was produced using an even RIN of 1234 and the encrypted code for the scrambled module format was produced using an odd RIN of 4321. When using either encrypted program "MAIN1E" or MAIN2E", when running the user's copy, the security module is called upon reaching "*messgl" or "*messg2" and the encrypted bits are sent out to the SPP and exclusive OR'd with the key (i.e., either "101 ... " or "0101 ... " depending on the RIN in use), and returned to the user's computer in decrypted form as the equivalent of "This is". Note that while a location-by-location comparison of "MAIN1" (unencrypted) and "MAIN1E" could reveal the encrypted locations, this type of comparison is rendered more difficult by scrambling the order of the modules as in "MAIN2E". In practice, it is intended that a longer random number PEK will be used and executable instructions as well as program data will be encrypted in a similar manner.

The foregoing system thus solves the problem of secure distribution of software to users by associating each unique copy with specific hardware to which the end user's computer must be connected. Copies of the user's program copy will only operate when the SPP with the right PEK and RIN is attached. When used in a phone line network, the system provides a powerful means of providing ongoing service to users. For example, the user can be notified of and provided with software enhancements via the network as soon as they are available. Moreover, the SPP provides for time-limited authorization. At the end of a trial period or rental term, the RIN for the borrowed software is cancelled, thus disabling further use.

Among the various other possible configurations of the present system are local area networks. Modem communication is not the essential embodiment of the invention, only the preferred one. The invention also lends itself to use as a terminal verifier. Instead of using a password, the SPP can be used to decrypt a code from a host computer and retransmit a decoded password to the host to verify authorization for access to secure data, for example.

Employing EEPROM's in the SPP opens up the possibility of downloading completely new software for running the SPP. Even new PEK's can be added by "remote control" from the SEC. Thus, the SEC maintains control over the cryptographic system in use by the SPP. For example, in addition to the exclusive OR algorithm, new algorithms with entirely different, perhaps more complex logic functions, could be added, including nonreversible keys.

While the software distribution system of the present invention has been described with reference to its preferred embodiments, various modifications and alterations in both hardware and software will occur to those skilled in the art from the foregoing detailed description and the accompanying drawings. These and other modifications and variations are intended to fall within the scope of the appended claims and equivalents thereto.

CLAIMS

1. A method of distributing software via an electronic communications network from a central facility with storage capacity for a library of available programs to a plurality of users' computers such that each distributed copy is usable only on specific user hardware, comprising the steps of

responding to a specific user request for a specific software program by generating a unique index code and preparing a unique user copy by encrypting selected passages of said program in a manner such that a given algorithm operating on said encrypted passage and a key specified by said index code and a user-specific master code will yield the plaintext version of said passage,

electronically transmitting said index code and said program with encrypted passages to the user, registering the index code in an independent auxiliary device interconnected with the user's computer,

storing the transmitted program with encrypted passages in the user's computer system on user selected media, when running the program with the encrypted passages on the user's computer, suspending normal execution at each encrypted passage and decrypting the encrypted passage by means of the auxiliary device by algorithmically combining the key specified by said index code and the user-specific master code with the encrypted passage and returning plaintext to said user's computer, and

continuing normal execution until e countering another encrypted passage,

whereby each user gets a different copy of the same program but no user ever has a complete plaintext version residing at any given time in the user's system memory so that each program copy is wedded to specific user hardware.

2. The method of claim 1, further comprising the step of

issuing differentiated independent auxiliary devices to said users having unique decryption master codes recorded at the central facility,

before preparing software for delivery, identifying the user's independent auxiliary device and looking up its decryption master code,

then preparing the unique copy by encrypting passages of the user selected program in a manner such that a given algorithm operating on (1) a key produced by a combination of the transmitted index code and the user's master code and (2) the encrypted passage will yield a plaintext version of the passage.

3. The method of claim 1, further comprising automatically removing the index code from the independent auxiliary device after a predetermined usage interval,

whereby the user's copy of the program is automatically disabled, for example, after a predetermined time interval.

4. The method of claim 2, wherein said issuing step includes factory loading each independent auxiliary device with a different decryption master code and recording each such master code at the central facility.

5. The method of claim 2, wherein the issuing step includes selecting the decryption master codes at the central facility after distribution of the independent auxiliary devices to the users and electronically transmitting a unique master code to each of the independent auxiliary devices upon its initial request for software.

6. The method of claim 5, wherein the step of electronically transmitting the decryption master code includes transmitting an encrypted version of the master code and decrypting the master code before storing it in the independent auxiliary device.

7. The method of claim 1, wherein at least some of the encrypted passages of the program are software instructions themselves.

8. The method of claim 1, further comprising the step of in some fashion scrambling the order of the modules in the user's copy before transmission to frustrate comparison with the original version of the program.

9. A software protection processor for an end user computer with a communications link to a central computer facility containing a software library, comprising.

means for storing a unique package encryption key (PEK),

means for receiving via said communications link and storing a registration index number (RIN) from the central facility uniquely associated with a specific software program to be stored in the end user's computer system,

logic means for modifying the PEK with the RIN to produce a specific decryption key,

computer means responsive to the presentation of encrypted data by the user's computer for decrypting said data by algorithmically combining it with the specific decryption key to produce a decrypted data output to said user's computer during program execution by the user's computer,

whereby a unique copy or software chosen by the user can be prepared by the central facility by encrypting selected passages of the software in a manner such that they can be decrypted by the software protection processor by algorithmically combining them with a decryption key produced by modifying the PEK with the RIN so that the user's copy will not run properly unless his computer is connected to a Software Protection Processor with the correct PEK and RIN.

10. The apparatus of claim 9, further comprising means for disabling the software specific RIN after a predetermined usage interval,

whereby the selected software is disabled after, for example, a predetermined trial period or rental term.

11. A data security apparatus for a user's computer having a communications link with a central computing facility, comprising

an independent auxiliary device electronically separate from but connected to the user's computer including

means for storing a unique first code,

means for receiving via said communications link a second unique code,

means for modifying said first code with said second code to produce a third code,

means responsive to the presentation of encrypted data for decrypting said data by algorithmically combining it with said third code,

whereby data presented over the communications link as an encrypted password, for example, or by the user's computer can be decrypted for verification.

12. The apparatus of claim 11, further comprising
means for disabling said second code after a predetermined usage interval.

13. A method of distributing software substantially as herein described with reference to the drawings.

14. Data security apparatus constructed and arranged substantially as herein described and shown in the drawings.

Printed in the United Kingdom for
Her Majesty's Stationery Office, Dd 8818935, 1985, 4235.
Published at The Patent Office, 25 Southampton Buildings,
London, WC2A 1AY, from which copies may be obtained.